

Dự đoán Khả năng Sống sót của Hành khách Titanic

Họ và tên: Nguyễn Thanh Việt

Email: ntviet@binhminhplastic.com.vn

Ngày hoàn thành: 07/07/2025

Mục lục

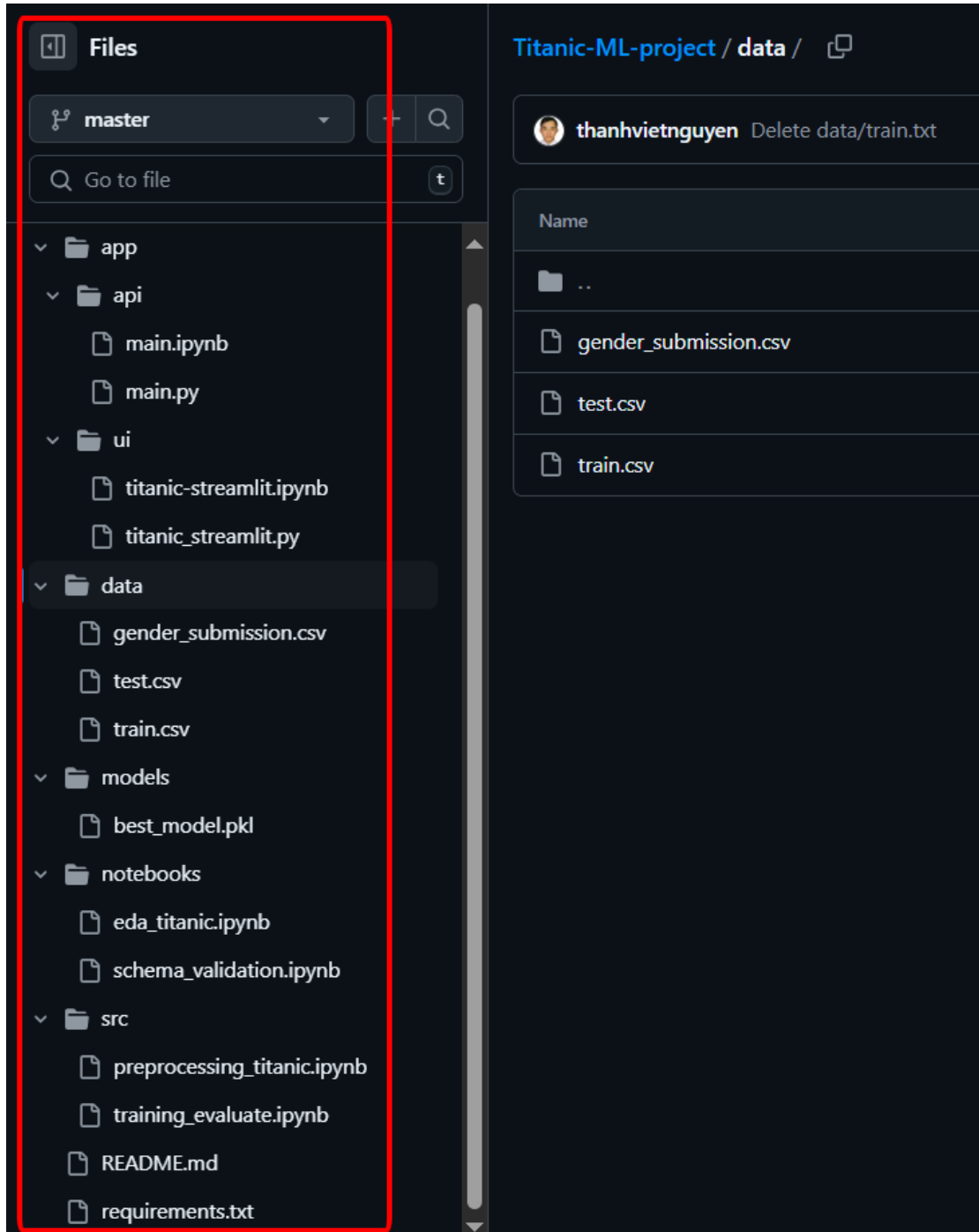
1. Giới thiệu bài toán
2. Phân tích và khám phá dữ liệu (EDA)
3. Tiền xử lý và kỹ thuật đặc trưng
4. Huấn luyện mô hình
5. Đánh giá mô hình
6. Triển khai hệ thống
7. Video demo
8. Hướng phát triển
9. Phụ lục

1. Giới thiệu bài toán

Bài toán dự đoán khả năng sống sót của hành khách trên tàu Titanic là một bài toán phân loại nhị phân cổ điển. Dựa vào thông tin cá nhân như tuổi, giới tính, hạng vé, giá vé,... hệ thống sẽ dự đoán hành khách có sống sót hay không.

2. Phân tích và khám phá dữ liệu (EDA)

Cấu trúc thư mục mã nguồn



Dữ liệu được lấy từ Kaggle Titanic Dataset. Tổng cộng có 891 hành khách với các thông tin như Pclass, Sex, Age, SibSp, Parch, Fare, Embarked.

eda_titanic.ipynb

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all Copy to Drive

```
[ ] 1 #Đọc dữ liệu
    2 df = pd.read_csv("https://raw.githubusercontent.com/thanhvietnguyen/Titanic-ML-project/refs/heads/master/data/train.csv")
```

```
[ ] 1 #Tổng quan dữ liệu
    2 print("Kích thước:", df.shape)
    3 display(df.head())
```

Kích thước: (891, 12)

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
1 display(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   PassengerId      891 non-null    int64
1   Survived         891 non-null    int64
2   Pclass           891 non-null    int64
3   Name             891 non-null    object
4   Sex              891 non-null    object
5   Age              714 non-null    float64
6   SibSp            891 non-null    int64
7   Parch            891 non-null    int64
8   Ticket           891 non-null    object
9   Fare             891 non-null    float64
10  Cabin            204 non-null    object
11  Embarked         889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

- Cột Age, Cabin và Embarked có giá trị thiếu.

eda_titanic.ipynb

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all Copy to Drive

```
[ ] 1 #Kiểm tra Missing Values
    2 missing = df.isnull().sum()
    3 print("Missing values:\n", missing)
```

Missing values:

PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687
Embarked	2
dtype:	int64

Vì cột Cabin có số lượng missing value quá lớn (hơn 77%) nên sẽ loại bỏ biến này khỏi các feature.

- Tỷ lệ sống sót của nữ cao hơn nam.



eda_titanic.ipynb

File Edit View Insert Runtime Tools Help

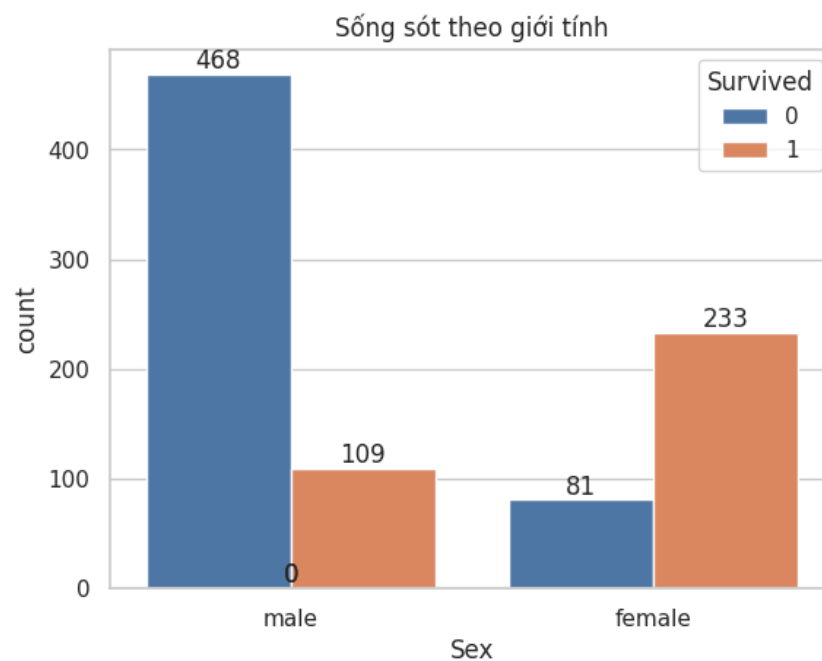
Commands

+ Code + Text

Run all

Copy to Drive

```
1 #Giới tính
2 ax = sns.countplot(x='Sex', hue='Survived', data=df)
3 for p in ax.patches:
4     height = p.get_height()
5     ax.annotate(f'{int(height)}',
6                 (p.get_x() + p.get_width()/2., height),
7                 ha='center', va='bottom',
8                 xytext=(0, 0),
9                 textcoords='offset points')
10 plt.title("Sống sót theo giới tính")
11 plt.show()
```



- Tỷ lệ sống sót theo lối lên tàu



eda_titanic.ipynb

File Edit View Insert Runtime Tools Help

Commands

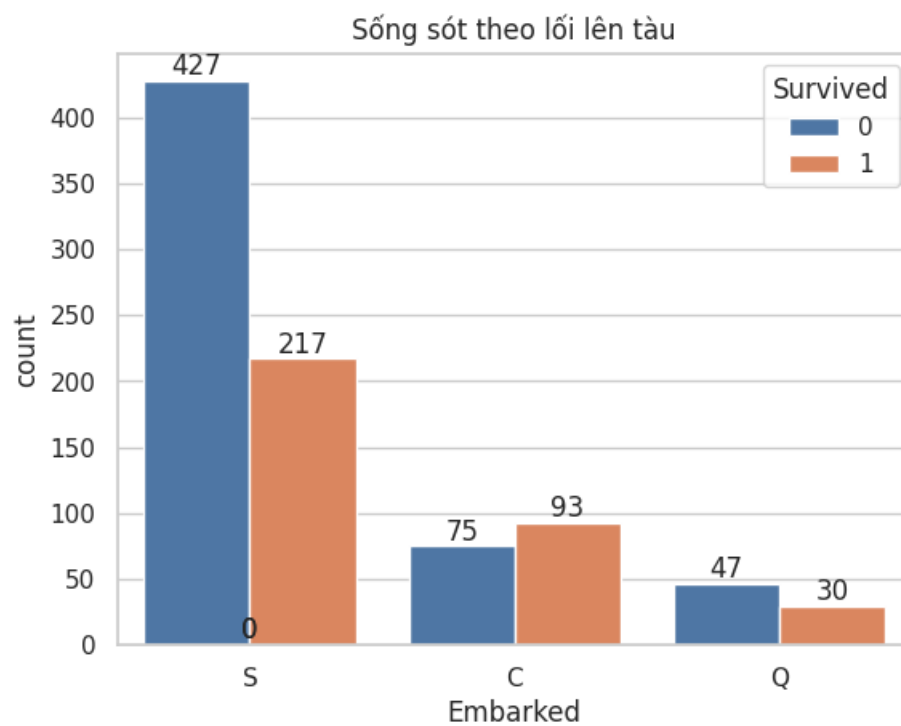
+ Code + Text

Run all

Copy to Drive



```
1 #Lối lên tàu
2 ax = sns.countplot(x='Embarked', hue='Survived', data=df)
3 for p in ax.patches:
4     height = p.get_height()
5     ax.annotate(f'{int(height)}',
6                 (p.get_x() + p.get_width()/2., height),
7                 ha='center', va='bottom',
8                 xytext=(0, 0),
9                 textcoords='offset points')
10 plt.title("Sống sót theo lối lên tàu")
11 plt.show()
```



- Tỷ lệ sống theo hạng vé



eda_titanic.ipynb

File Edit View Insert Runtime Tools Help

Q Commands

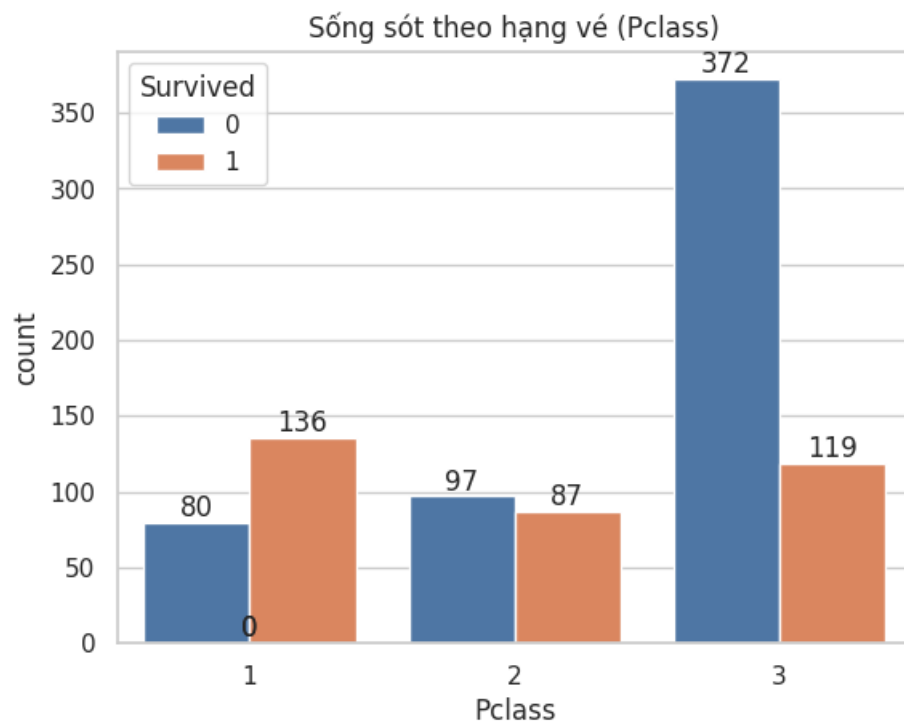
+ Code + Text

▶ Run all ▼

Copy to Drive



```
1 #Hạng vé
2 ax = sns.countplot(x='Pclass', hue='Survived', data=df)
3 for p in ax.patches:
4     height = p.get_height()
5     ax.annotate(f'{int(height)}',
6                 (p.get_x() + p.get_width()/2., height),
7                 ha='center', va='bottom',
8                 xytext=(0, 0),
9                 textcoords='offset points')
10 plt.title("Sống sót theo hạng vé (Pclass)")
11 plt.show()
```



- Phân phối tuổi tập trung ở khoảng 20–40 tuổi.



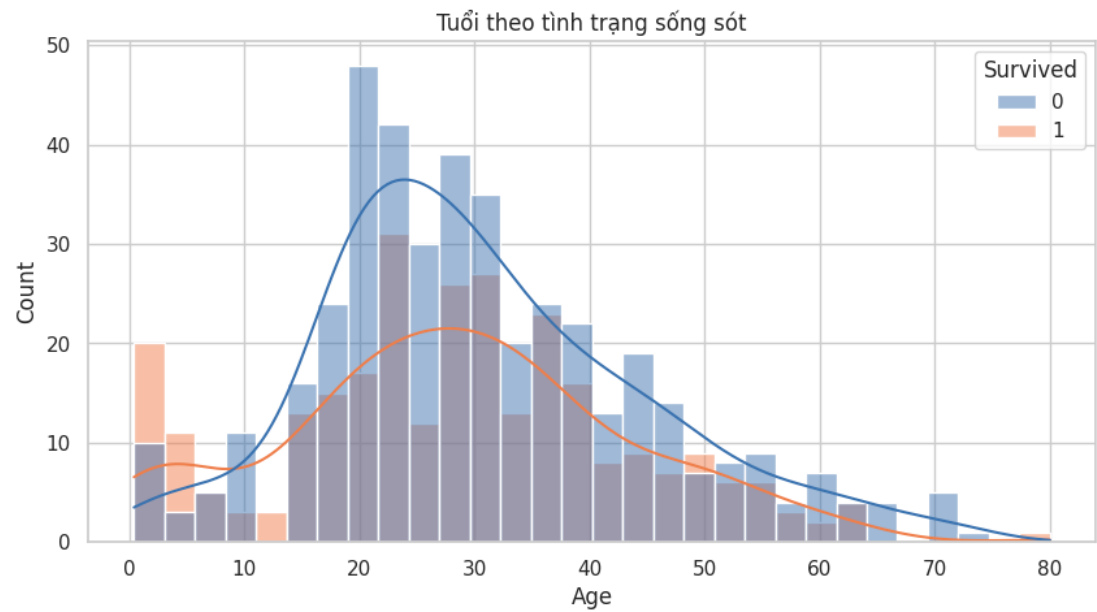
eda_titanic.ipynb

File Edit View Insert Runtime Tools Help

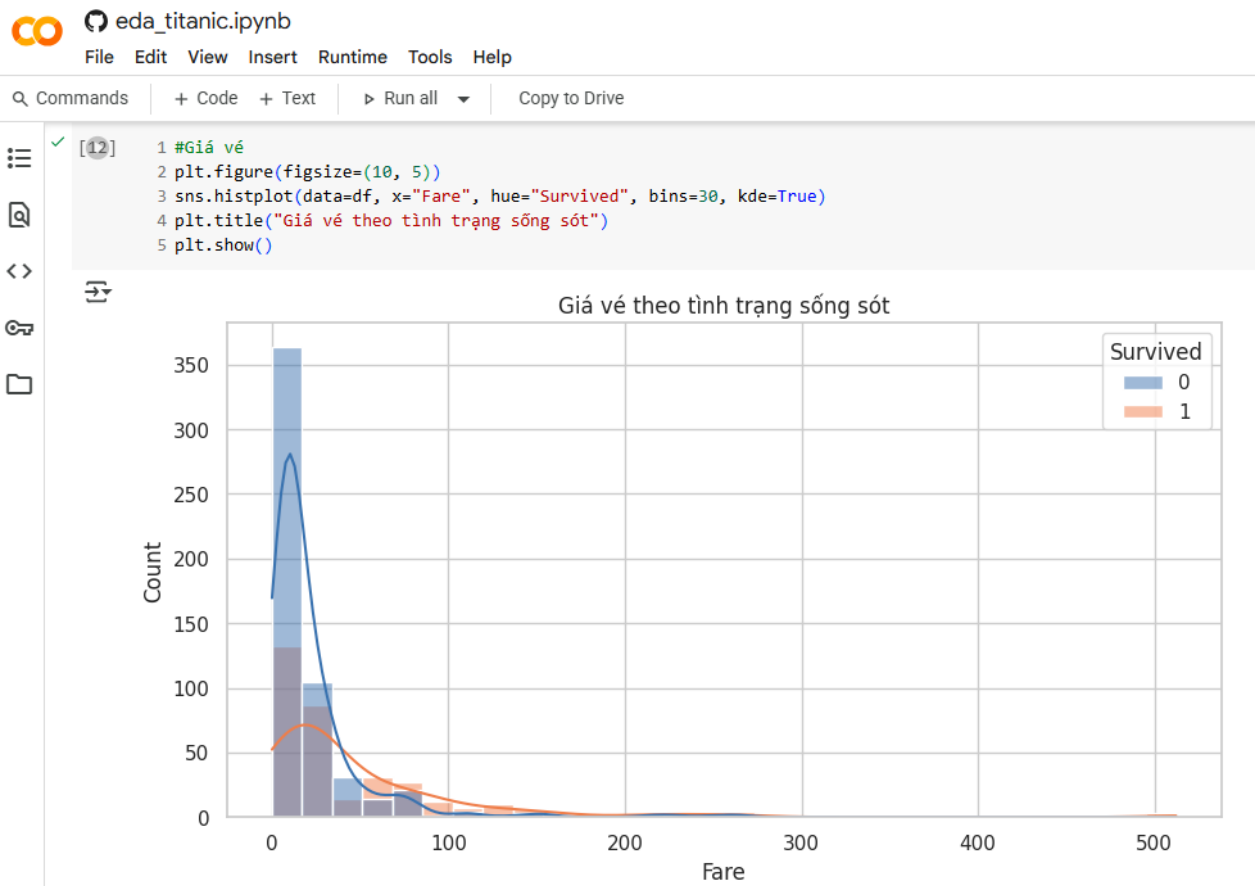
Commands + Code + Text Run all Copy to Drive



```
1 #Tuổi
2 plt.figure(figsize=(10, 5))
3 sns.histplot(data=df, x="Age", hue="Survived", bins=30, kde=True)
4 plt.title("Tuổi theo tình trạng sống sót")
5 plt.show()
```



- Phân phối giá vé



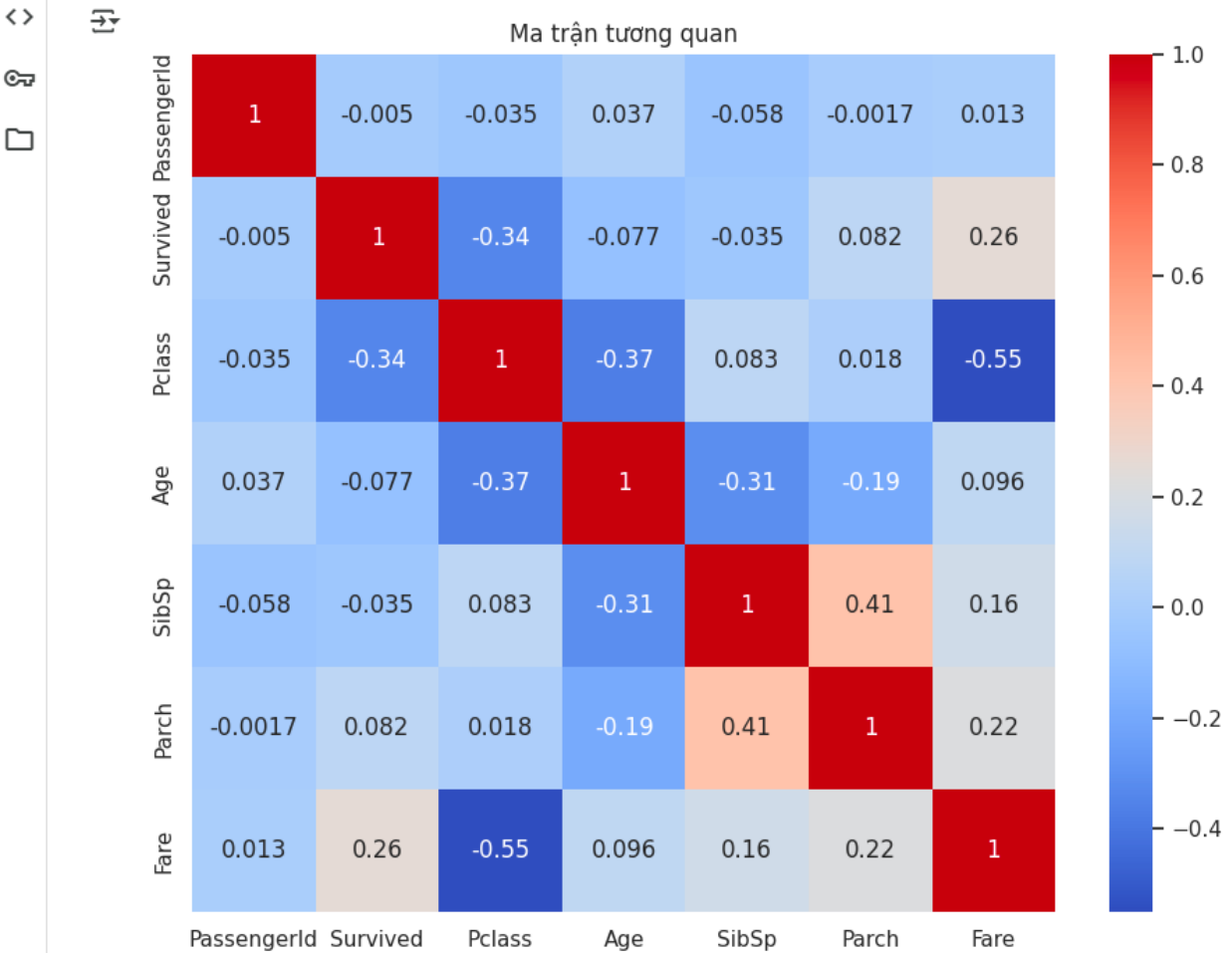
- Mối tương quan giữa các biến số numeric

eda_titanic.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

```
[ ] 1 #Mối tương quan giữa các biến số numeric  
2 plt.figure(figsize=(10, 8))  
3 sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')  
4 plt.title("Ma trận tương quan")  
5 plt.show()
```



- Tạo biến mới Quy mô gia đình



→ Xác định được 6 feature là: Giới tính (Sex), Lối lên tàu (Embarked), Tuổi (Age), Hạng vé (Pclass), Giá vé (Fare) và Quy mô gia đình (FamilySize).

3. Tiền xử lý và kỹ thuật đặc trưng

- Tạo đặc trưng mới: $\text{FamilySize} = \text{SibSp} + \text{Parch} + 1$.

```
preprocessing_titanic.ipynb
File Edit View Insert Runtime Tools Help

[ ] 1 import pandas as pd
    2 import numpy as np
    3 from sklearn.impute import SimpleImputer
    4 from sklearn.preprocessing import OneHotEncoder, StandardScaler
    5 from sklearn.pipeline import Pipeline
    6 from sklearn.compose import ColumnTransformer

[ ] 1 # Đọc dữ liệu
    2 df = pd.read_csv("https://raw.githubusercontent.com/thanhvietnguyen/Titanic-ML-project/refs/heads/master/data/train.csv")

[ ] 1 #1 - Tạo biến đặc trưng mới
    2 #Quy mô gia đình = Người thân đi cùng(ace/vc(SibSp) + cha mẹ/con(Parch)) + 1 (bản thân)
    3 df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
```

- Xác định các biến đầu vào và biến mục tiêu.

```
preprocessing_titanic.ipynb
File Edit View Insert Runtime Tools Help

[ ] 1 #2 - Xác định biến đầu vào và biến mục tiêu
    2 #6 biến đầu vào: Tuổi, Giá vé, Quy mô gia đình, Giới tính, Nơi lên tàu, Hạng vé
    3 #Biến mục tiêu: Sống hay chết
    4 X = df[["Age", "Fare", "FamilySize", "Sex", "Embarked", "Pclass"]]
    5 y = df["Survived"]
```

- Chia các biến thành nhóm.

```
preprocessing_titanic.ipynb
File Edit View Insert Runtime Tools Help

[ ] 1 #3 - Chia các biến thành nhóm
    2 numeric_features = ["Age", "Fare", "FamilySize"]
    3 categorical_features = ["Sex", "Embarked", "Pclass"]
```

- Xử lý điền thiếu nhóm số bằng chuẩn hoá, nhóm phân loại bằng one-hot encode.

```
preprocessing_titanic.ipynb
File Edit View Insert Runtime Tools Help

[ ] 1 #4 - Pipeline xử lý từng nhóm biến
     2 # Pipeline xử lý số: điền thiếu --> chuẩn hóa
     3 numeric_pipeline = Pipeline([
     4     ('imputer', SimpleImputer(strategy="median")),
     5     ('scaler', StandardScaler())
     6 ])

[ ] 1 # Pipeline xử lý phân loại: điền thiếu --> One-hot encode
     2 categorical_pipeline = Pipeline([
     3     ('imputer', SimpleImputer(strategy="most_frequent")),
     4     ('encoder', OneHotEncoder(handle_unknown='ignore'))
     5 ])
```

- Kết hợp 2 pipeline

```
preprocessing_titanic.ipynb
File Edit View Insert Runtime Tools Help

[ ] 3 numeric_pipeline = Pipeline([
     4     ('imputer', SimpleImputer(strategy="median")),
     5     ('scaler', StandardScaler())
     6 ])

[ ] 1 # Pipeline xử lý phân loại: điền thiếu --> One-hot encode
     2 categorical_pipeline = Pipeline([
     3     ('imputer', SimpleImputer(strategy="most_frequent")),
     4     ('encoder', OneHotEncoder(handle_unknown='ignore'))
     5 ])
```

```
[ ] 1 # Kết hợp 2 pipeline lại
     2 preprocessor = ColumnTransformer([
     3     ('num', numeric_pipeline, numeric_features),
     4     ('cat', categorical_pipeline, categorical_features)
     5 ])
```

4. Huấn luyện mô hình

Hai mô hình được sử dụng là Logistic Regression và Random Forest.

- Chia dữ liệu 80% train, 20% test.

```
training_evaluate.ipynb
File Edit View Insert Runtime Tools Help

Q Commands | + Code + Text | ▶ Run all | Copy to Drive

Model Training

[ ] 1 import pandas as pd
    2 import numpy as np
    3
    4 from sklearn.linear_model import LogisticRegression
    5 from sklearn.ensemble import RandomForestClassifier
    6 from sklearn.model_selection import train_test_split
    7 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
    8
    9 import joblib

[ ] 1 #Đọc và chuẩn bị dữ liệu
    2 df = pd.read_csv("https://raw.githubusercontent.com/thanhvietnguyen/Titanic-ML-project/refs/heads/master/data/train.csv")
    3 df['FamilySize'] = df['SibSp'] + df['Parch'] + 1 #New feature: Quy mô gia đình = Số người thân đi cùng + bản thân
    4 X = df[['Age', 'Fare', 'FamilySize', 'Sex', 'Embarked', 'Pclass']] #6 biến đầu vào
    5 y = df["Survived"]

▶ 1 #Tiền xử lý
    2 from sklearn.pipeline import Pipeline
    3 from sklearn.compose import ColumnTransformer
    4 from sklearn.preprocessing import OneHotEncoder, StandardScaler
    5 from sklearn.impute import SimpleImputer
    6
    7 numeric_features = ["Age", "Fare", "FamilySize"]
    8 categorical_features = ["Sex", "Embarked", "Pclass"]
    9
    10 numeric_pipeline = Pipeline([
    11     ('imputer', SimpleImputer(strategy="median")),
    12     ('scaler', StandardScaler())
    13 ])
    14
    15 categorical_pipeline = Pipeline([
    16     ('imputer', SimpleImputer(strategy="most_frequent")),
    17     ('encoder', OneHotEncoder(handle_unknown='ignore'))
    18 ])
```

```
training_evaluate.ipynb
File Edit View Insert Runtime Tools Help

Q Commands | + Code + Text | ▶ Run all | Copy to Drive

[ ] 17 ('encoder', OneHotEncoder(handle_unknown='ignore'))
    18 ])
    19
    20 preprocessor = ColumnTransformer([
    21     ('num', numeric_pipeline, numeric_features),
    22     ('cat', categorical_pipeline, categorical_features)
    23 ])

[ ] 1 #Chia tập train/test 80/20
    2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Train 2 mô hình Logistic Regression và Random Forest

```
training_evaluate.ipynb
File Edit View Insert Runtime Tools Help

Q Commands | + Code + Text | ▶ Run all | Copy to Drive

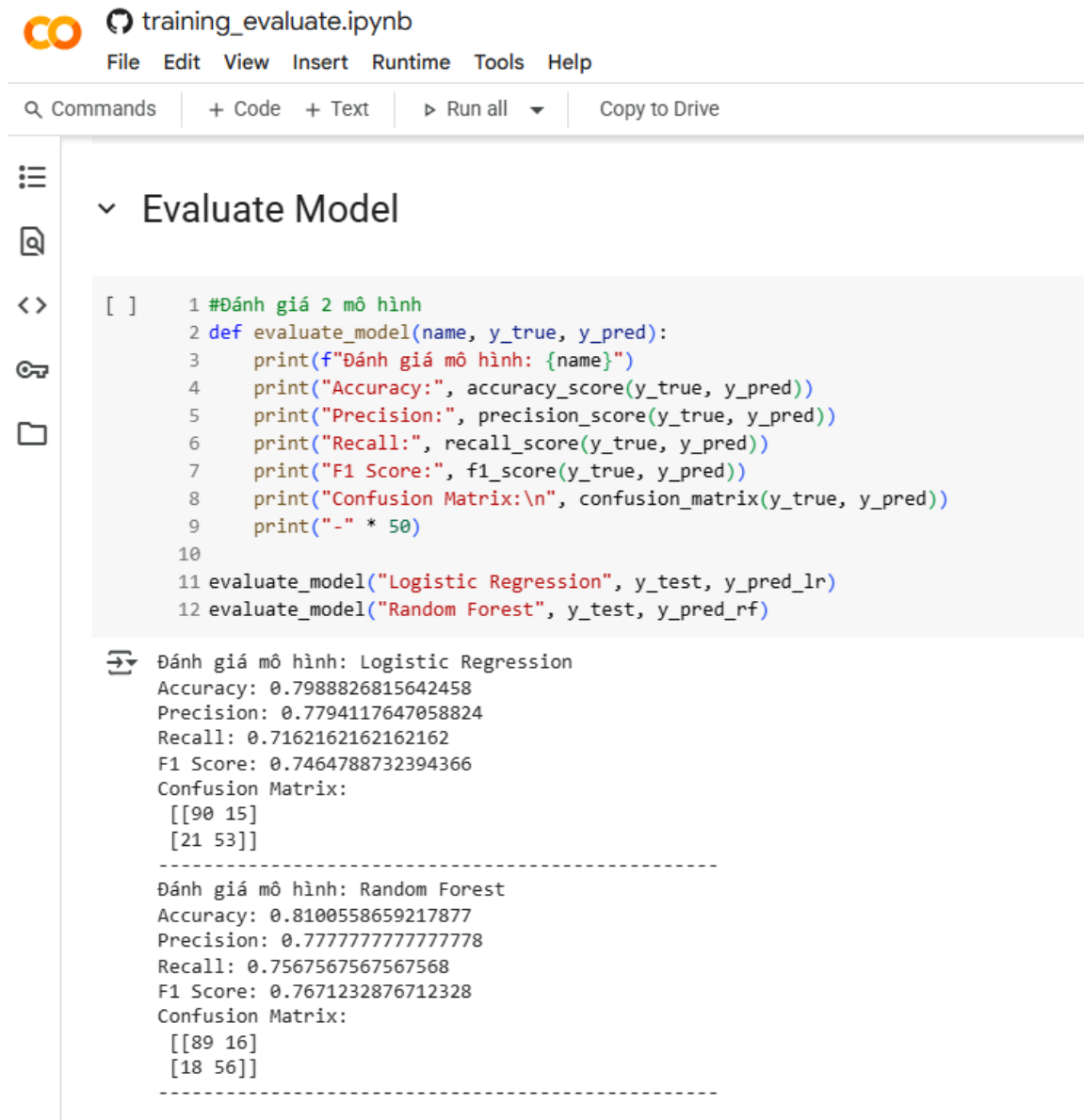
[ ] 1 #Train Logistic Regression
    2 from sklearn.pipeline import make_pipeline
    3
    4 logreg_model = make_pipeline(preprocessor, LogisticRegression(max_iter=1000))
    5 logreg_model.fit(X_train, y_train)
    6 y_pred_lr = logreg_model.predict(X_test)

[ ] 1 #Train Random Forest
    2 rf_model = make_pipeline(preprocessor, RandomForestClassifier(n_estimators=100, random_state=42))
    3 rf_model.fit(X_train, y_train)
    4 y_pred_rf = rf_model.predict(X_test)
```

5. Đánh giá mô hình

Kết quả đánh giá:

- Logistic Regression: Accuracy 79.89%, Precision 77.94%, Recall 71.62%, F1-Score 74.65%
- Random Forest: Accuracy 81%, Precision 77.78%, Recall 75.68%, F1-Score 76.71%



The screenshot shows a Jupyter Notebook titled "training_evaluate.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with "Commands", "+ Code", "+ Text", "Run all", and "Copy to Drive". The left sidebar contains icons for a table of contents, search, code editor, key, and file explorer. The main area displays a code cell titled "Evaluate Model" with the following Python code:

```
[ ] 1 #Đánh giá 2 mô hình
2 def evaluate_model(name, y_true, y_pred):
3     print(f"Đánh giá mô hình: {name}")
4     print("Accuracy:", accuracy_score(y_true, y_pred))
5     print("Precision:", precision_score(y_true, y_pred))
6     print("Recall:", recall_score(y_true, y_pred))
7     print("F1 Score:", f1_score(y_true, y_pred))
8     print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
9     print("-" * 50)
10
11 evaluate_model("Logistic Regression", y_test, y_pred_lr)
12 evaluate_model("Random Forest", y_test, y_pred_rf)
```

The output of the code cell shows the evaluation results for two models:

```
Đánh giá mô hình: Logistic Regression
Accuracy: 0.7988826815642458
Precision: 0.7794117647058824
Recall: 0.7162162162162162
F1 Score: 0.7464788732394366
Confusion Matrix:
[[90 15]
 [21 53]]
-----
Đánh giá mô hình: Random Forest
Accuracy: 0.8100558659217877
Precision: 0.7777777777777778
Recall: 0.7567567567567568
F1 Score: 0.7671232876712328
Confusion Matrix:
[[89 16]
 [18 56]]
-----
```



Evaluation Charts



```
[ ] 1 import matplotlib.pyplot as plt
    2 import seaborn as sns
    3 from sklearn.metrics import confusion_matrix
```

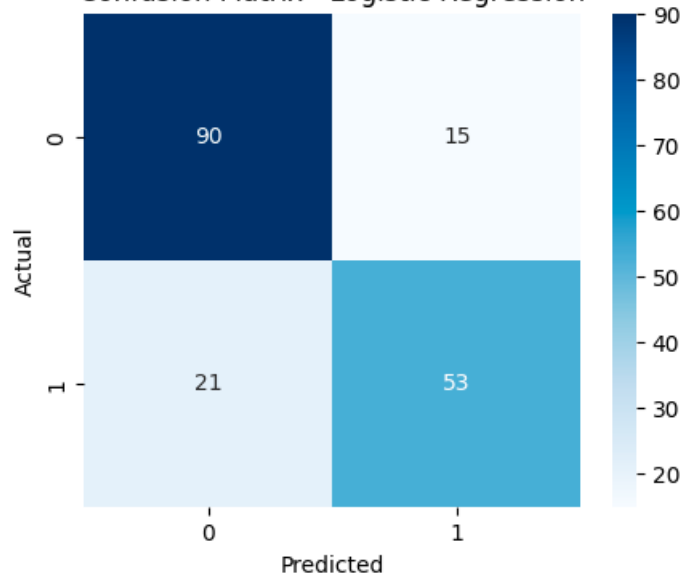
```
[ ] 1 def plot_confusion(y_true, y_pred, title="Confusion Matrix"):
    2     cm = confusion_matrix(y_true, y_pred)
    3     plt.figure(figsize=(5,4))
    4     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    5     plt.title(title)
    6     plt.xlabel("Predicted")
    7     plt.ylabel("Actual")
    8     plt.show()
```

```
[ ] 1 #Dự đoán xác suất
    2 y_proba_lr = logreg_model.predict_proba(X_test)[: , 1]
    3 y_proba_rf = rf_model.predict_proba(X_test)[: , 1]
```

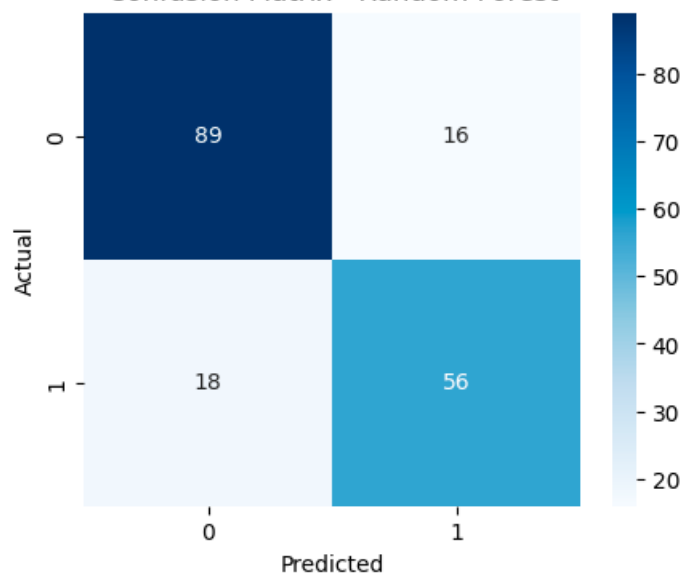
```
▶ 1 #Confusion Matrix
    2 plot_confusion(y_test, y_pred_lr, "Confusion Matrix - Logistic Regression")
    3 plot_confusion(y_test, y_pred_rf, "Confusion Matrix - Random Forest")
```



Confusion Matrix - Logistic Regression



Confusion Matrix - Random Forest



- Random Forest cho kết quả tốt hơn → Chọn Random Forest để triển khai.
- Lưu mô hình tốt nhất: Random Forest

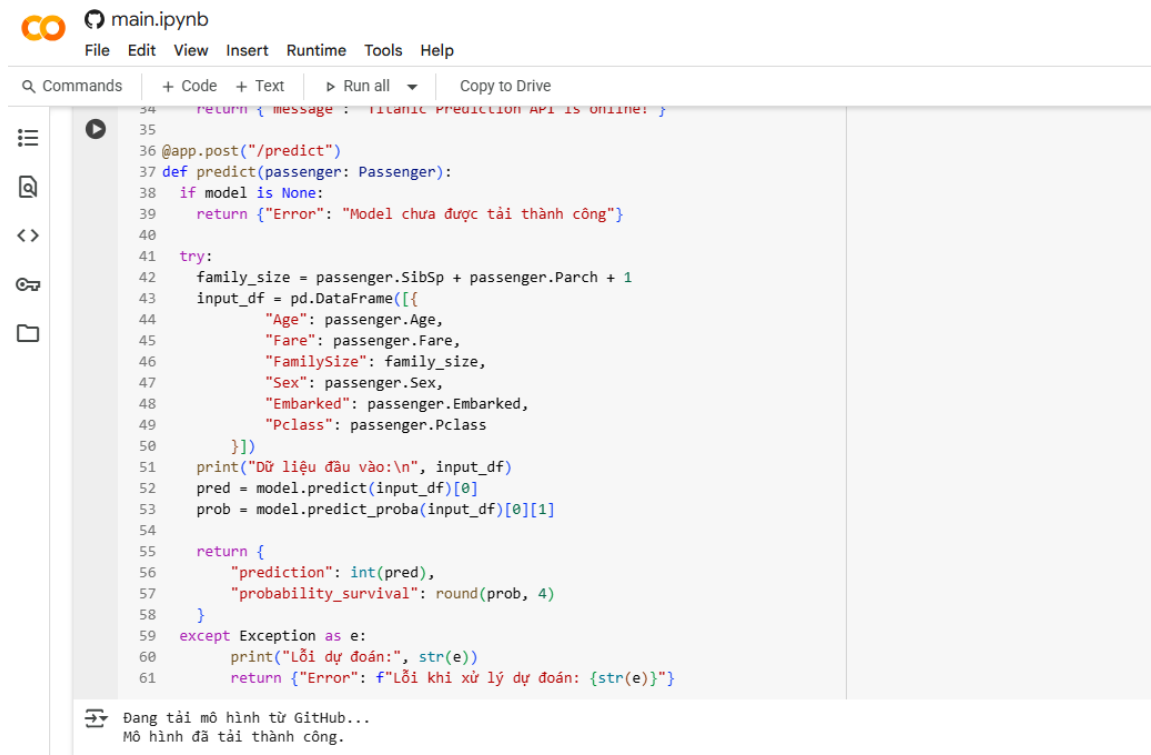

```
[ ] 1 # Lưu mô hình tốt nhất
    2 joblib.dump(rf_model, "best_model.pkl")
    3 print("Đã lưu mô hình tốt nhất vào best_model.pkl")
```

↗ Đã lưu mô hình tốt nhất vào best_model.pkl

6. Triển khai hệ thống

- API: sử dụng FastAPI

```
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3 import pandas as pd
4 import joblib
5 # Using the BytesIO you create a file object out of the response that you get from GitHub
6 from io import BytesIO
7 import requests
8
9 app = FastAPI()
10
11 # Load mô hình
12 mlink = 'https://github.com/thanhvietnguyen/Titanic-ML-project/blob/master/models/best_model.pkl?raw=true'
13 try:
14     print("Đang tải mô hình từ GitHub...")
15     response = requests.get(mlink)
16     response.raise_for_status() # Gây lỗi nếu không=200
17     mfile = BytesIO(response.content)
18     model = joblib.load(mfile)
19     print("Mô hình đã tải thành công.")
20 except Exception as e:
21     print("Lỗi khi tải mô hình:", str(e))
22
23 class Passenger(BaseModel):
24     Pclass: int
25     Sex: str
26     Age: float
27     SibSp: int
28     Parch: int
29     Fare: float
30     Embarked: str
31
32 @app.get("/")
33 def read_root():
34     return {"message": "Titanic Prediction API is online!"}
35
36 @app.post("/predict")
37 def predict(passenger: Passenger):
38     if model is None:
39         return {"Error": "Model chưa được tải thành công"}
40
```

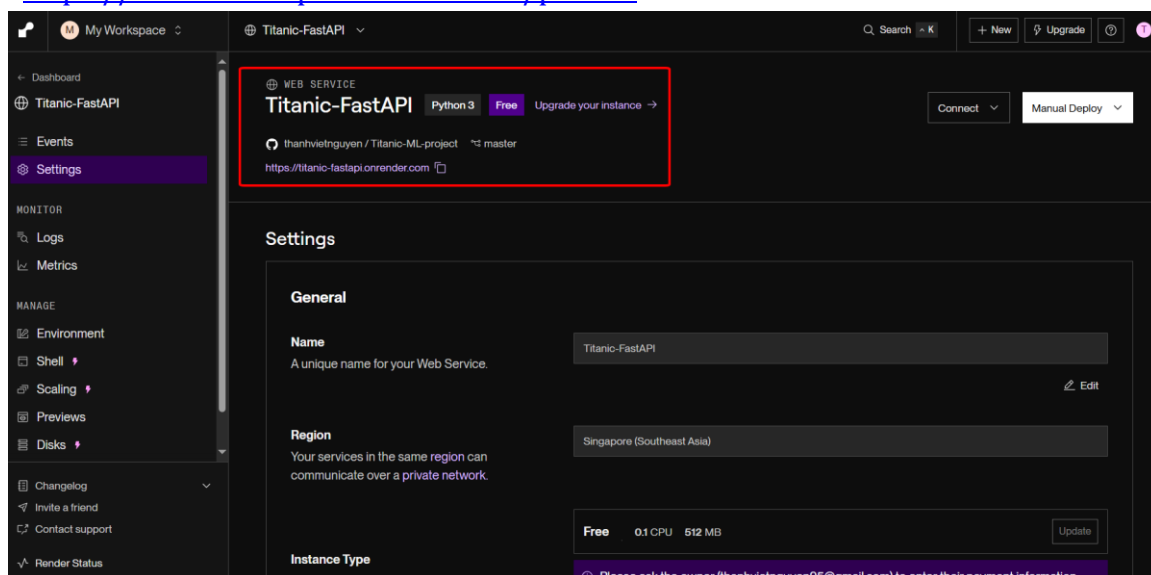


```
34 return { "message": "Titanic Prediction API is Online!" }
35
36 @app.post("/predict")
37 def predict(passenger: Passenger):
38     if model is None:
39         return {"Error": "Model chưa được tải thành công"}
40
41     try:
42         family_size = passenger.SibSp + passenger.Parch + 1
43         input_df = pd.DataFrame([
44             "Age": passenger.Age,
45             "Fare": passenger.Fare,
46             "FamilySize": family_size,
47             "Sex": passenger.Sex,
48             "Embarked": passenger.Embarked,
49             "Pclass": passenger.Pclass
50         ])
51         print("Dữ liệu đầu vào:\n", input_df)
52         pred = model.predict(input_df)[0]
53         prob = model.predict_proba(input_df)[0][1]
54
55         return {
56             "prediction": int(pred),
57             "probability_survival": round(prob, 4)
58         }
59     except Exception as e:
60         print("Lỗi dự đoán:", str(e))
61         return {"Error": f"Lỗi khi xử lý dự đoán: {str(e)}"}
```

Đang tải mô hình từ GitHub...
Mô hình đã tải thành công.

- Deploy trên Render.com tại URL:

<https://titanic-fastapi.onrender.com/predict>



The first screenshot shows the 'Build & Deploy' configuration page for the 'Titanic-FastAPI' service. The repository is set to 'https://github.com/thanhvietnguyen/Titanic-ML-project', the branch is 'master', and the Git credentials are 'thanhvietnguyen95@gmail.com (you)'. The root directory is optional and currently empty.


The second screenshot shows the 'Build Command' configuration page. The build command is '\$ pip install -r requirements.txt'. The pre-deploy command is optional and empty. The start command is '\$ PYTHONPATH=./app uvicorn api.main:app --host 0.0.0.0 --port 10000'. The auto-deploy is set to 'On Commit'. The deploy hook is a secret URL.

The third screenshot shows the deployment status and logs. A message indicates that the free instance will spin down with inactivity. The deployment is 'Live' and was created on 'July 3, 2025 at 5:05 PM'. The logs show the application startup process, including the uvicorn command and the service being available at 'https://titanic-fastapi.onrender.com'.

- Giao diện: sử dụng Streamlit, deploy trên Streamlit Cloud tại URL:
<https://ntv-titanic-ml-projecttv.streamlit.app/>








```
1 import streamlit as st
2 import requests
3
4 st.title("Dự đoán sống sót Titanic")
5 st.write("Nhập thông tin hành khách và nhận dự đoán liệu họ có sống sót hay không.")
6
7 API_URL = "https://titanic-fastapi.onrender.com/predict"
8
9 # Nhập dữ liệu từ người dùng
10 sex = st.selectbox("Giới tính", ["male", "female"])
11 pclass = st.selectbox("Hạng vé (Pclass)", [1, 2, 3])
12 age = st.slider("Tuổi", min_value=0, max_value=100, value=30)
13 sibsp = st.number_input("Số anh chị em/vợ chồng đi cùng (SibSp)", min_value=0, max_value=10, value=0)
14 parch = st.number_input("Số cha mẹ/con đi cùng (Parch)", min_value=0, max_value=10, value=0)
15 fare = st.number_input("Giá vé (Fare)", min_value=0.0, value=30.0)
16 embarked = st.selectbox("Nơi lên tàu (Embarked)", ["C", "Q", "S"])
17
18 # Gửi yêu cầu đến FastAPI
19 if st.button("Dự đoán"):
20     input_data = {
21         "Pclass": pclass,
22         "Sex": sex,
23         "Age": age,
24         "SibSp": sibsp,
25         "Parch": parch,
26         "Fare": fare,
27         "Embarked": embarked
28     }
29
30     try:
31         response = requests.post(API_URL, json=input_data)
32
33         st.write("Status code:", response.status_code)
34         st.write("Raw response:", response.text)
35
36         if response.status_code == 200:
37             result = response.json()
38
39             if "error" in result:
40                 st.error(result["error"])
41             else:
```


 titanic-streamlit.ipynb


File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼ Copy to Drive



```
26     "Fare": fare,
27     "Embarked": embarked
28 }
29
30 try:
31     response = requests.post(API_URL, json=input_data)
32
33     st.write("Status code:", response.status_code)
34     st.write("Raw response:", response.text)
35
36     if response.status_code == 200:
37         result = response.json()
38
39         if "error" in result:
40             st.error(result["error"])
41         else:
42             label = "Sống sót" if result["prediction"] == 1 else "Không sống sót"
43             st.success(f"Dự đoán: {label}")
44             st.info(f"Xác suất sống: {result['probability_survival'] * 100:.2f}%")
45         else:
46             st.error("Lỗi từ API: " + response.text)
47
48 except Exception as e:
49     st.error(f"Lỗi khi gọi API: {e}")
50
```

 2025-07-04 01:58:57.079 WARNING streamlit.runtime.scriptrunner_utils.script_run_context: Thread 'MainThread':
2025-07-04 01:58:57.215
Warning: to view this Streamlit app on a browser, run it with the following
command:

 thanhvietnguyen ▼


My apps

My profile

Explore

Discuss ↗

thanhvietnguyen's apps

 titanic-ml-project · master · app/ui/titanic_streamlit.py

- Người dùng nhập thông tin và nhận kết quả từ API.

Note: Khi sử dụng app lần đầu, nhấn Dự đoán thì có thể ứng dụng sẽ trả kết quả lâu vì API trên Render có thể đang Sleep nên phải chờ một lúc để API wake-up.

7. Video demo

Trong file đính kèm.

8. Hướng phát triển

- Dùng Keras hoặc PyTorch để thử mạng nơ-ron.
- Cho phép người dùng upload file CSV để dự đoán hàng loạt.
- Tối ưu mô hình bằng GridSearchCV.

9. Phụ lục

- Link GitHub: [thanhvietnguyen/Titanic-ML-project](https://github.com/thanhvietnguyen/Titanic-ML-project)
- Tham khảo: scikit-learn, FastAPI, Streamlit, Kaggle Titanic Dataset